

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-281912

(43)Date of publication of application : 27.10.1995

(51)Int.Cl.

G06F 11/00

G06F 9/34

G06F 9/42

(21)Application number : 06-069628

(71)Applicant : NIPPONDENSO CO LTD

(22)Date of filing : 07.04.1994

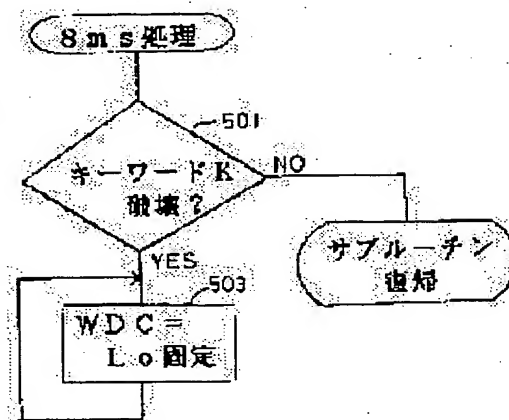
(72)Inventor : KAMIYA TSUTOMU
KONDO HIROSHI

(54) STACK ABNORMALITY DETECTING DEVICE

(57)Abstract:

PURPOSE: To surely detect the overflow of a stack pointer without applying the large load to a CPU.

CONSTITUTION: It is decided at initialization whether or not a key word K written in a key word area A0 of an address that is lower than a stack area by one degree is different from the contents of the present key word area (501). If not different, the processing returns directly to the original processing. If different, a signal WDC is fixed at a low level (503). A monitor part detects this state and forcibly resets a CPU. Thus the CPU is started again to carry out the processing in a normal state. Thus it is possible to prevent such a case where a stack pointer moves into an address lower than the area A0 to cause the runaway of a program or to destroy a RAM. Furthermore even an instantaneous stack abnormality can be detected since it remains as the destruction of the word K.



LEGAL STATUS

[Date of request for examination] 01.12.2000

[Date of sending the examiner's decision of rejection] 28.08.2001

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-281912

(43) 公開日 平成7年(1995)10月27日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 11/00	3 1 0 H			
9/34	3 4 0 C			
9/42	3 3 0 C			

審査請求 未請求 請求項の数5 O L (全 7 頁)

(21) 出願番号 特願平6-69628

(22) 出願日 平成6年(1994)4月7日

(71) 出願人 000004260

日本電装株式会社

愛知県刈谷市昭和町1丁目1番地

(72) 発明者 神谷 努

愛知県刈谷市昭和町1丁目1番地 日本電装株式会社内

(72) 発明者 近藤 浩

愛知県刈谷市昭和町1丁目1番地 日本電装株式会社内

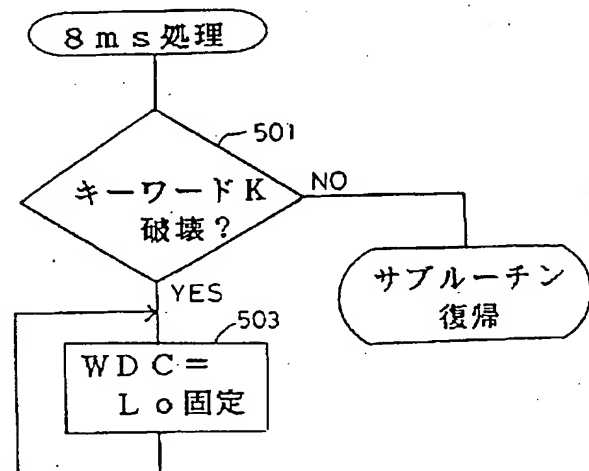
(74) 代理人 弁理士 足立 勉

(54) 【発明の名称】 スタック異常検出装置

(57) 【要約】

【目的】 CPUに大きな負荷をかけることなく、スタックポインタがオーバーフローを生じた場合を確実に捉える。

【構成】 イニシャル時にスタック領域より一つ低いアドレスのキーワード領域に書き込まれたキーワードKと、現在のキーワード領域の内容と異なるかどうかを判別する(ステップ501)。異なる場合はこのまま元の処理に復帰する。異なる場合は、WDC信号を低レベルに固定する(ステップ503)。監視部39がこの状態を検出してCPUを強制リセットする。従って、CPUは再度立ち上げ直されて正常な状態で処理を開始することになる。このように、キーワード領域A0より低いアドレスにスタックポインタが進んでプログラムが暴走したり、RAMが破壊されることを防止できる。しかも瞬間的なスタック異常が生じていてもキーワードKの破壊として残存しているのでその検出が可能となった。



(2)

【特許請求の範囲】

【請求項1】 プロセッサが使用するスタック領域のオーバーフローを検出するスタック異常検出装置であって、メモリ上のスタック領域に隣接するあるいは近傍のアドレスに設けられ、所定キーワードが設定されたキーワード領域と、
上記キーワード領域の設定内容が上記所定キーワードと同一か否かを監視し、同一でない場合はスタック異常と判定する判定手段と、
を備えたことを特徴とするスタック異常検出装置。

【請求項2】 上記キーワード領域が、スタック領域の最高点であるアドレスを越えたアドレスに設けられた請求項1記載のスタック異常検出装置。

【請求項3】 上記所定キーワードが、正常時にスタック領域に設定される可能性が低いまたは無い値である請求項1または2記載のスタック異常検出装置。

【請求項4】 上記所定キーワードが、スタック領域の最高点であるアドレスの次のアドレスに設けられている請求項2または3記載のスタック異常検出装置。

【請求項5】 上記キーワード領域が、複数のアドレスにわたって設けられている請求項1～4のいずれか記載のスタック異常検出装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、スタック異常検出装置に関し、特にコンピュータにおけるプログラムの実行に伴うスタックオーバーフローの検出装置に関する。

【0002】

【従来の技術】 マイクロコンピュータ等において、外来ノイズによりCPUの誤動作が発生した場合、スタックポインタの進みすぎ、いわゆるスタックオーバーフローにより、他領域のメモリ内容を破壊する可能性がある。一般に、スタックオーバーフローの異常を検出する方法として、スタック領域の上限アドレスと実際のアドレス（スタックポインタの値）とを比較し、実際のアドレスがこの上限アドレスを越えた時、スタックオーバーフローと判断するものがある（例えば、特開平3-6618号）。

【0003】

【発明が解決しようとする課題】 しかしながら、上記技術は、ソフトウェアで実現する場合、CPUが一つのシステムでは、当然、常時連続モニタ不可能であるため、瞬間的にスタックポインタのアドレスが上限値を超えた時は異常検出できないという問題が有った。また、常時連続モニタを可能にするにはハードウェアによる手段が必要となるため、ソフトウェアで実現する場合と比べてコストが増加するという問題が有った。

【0004】 この他に、スタックポインタが変化する状況、即ち、ソフトウェア割込やハードウェア割込が生じた際に、スタックポインタの値が異常な値を示していないかを検出する技術が提案されている（特開平2-29

3939号）。しかし、この技術も割込処理が発生するたびに、スタックポインタチェック処理を実施するため、CPUに与える負荷が大きくなり、他の処理速度に影響するため採用し難いものであった。

【0005】 この様な問題点に鑑み、本発明は、CPUに大きな負荷をかけることなく、かつスタックポインタがオーバーフローを生じた場合を確実に捉えることを目的としてなされたものである。

【0006】

【課題を解決するための手段】 請求項1記載の発明は、プロセッサが使用するスタック領域のオーバーフローを検出するスタック異常検出装置であって、メモリ上のスタック領域に隣接するあるいは近傍のアドレスに設けられ、所定キーワードが設定されたキーワード領域と、上記キーワード領域の設定内容が上記所定キーワードと同一か否かを監視し、同一でない場合はスタック異常と判定する判定手段と、を備えたことを特徴とするスタック異常検出装置である。

【0007】 請求項2記載の発明は、上記キーワード領域が、スタック領域の最高点であるアドレスを越えたアドレスに設けられた請求項1記載のスタック異常検出装置である。請求項3記載の発明は、上記所定キーワードが、正常時にスタック領域に設定される可能性が低いまたは無い値である請求項1または2記載のスタック異常検出装置である。

【0008】 請求項4記載の発明は、上記所定キーワードが、スタック領域の最高点であるアドレスの次のアドレスに設けられている請求項2または3記載のスタック異常検出装置である。請求項5記載の発明は、上記キーワード領域が、複数のアドレスにわたって設けられている請求項1～4のいずれか記載のスタック異常検出装置である。

【0009】

【作用及び発明の効果】 請求項1記載の発明は、メモリ上のスタック領域に隣接するあるいは近傍のアドレスにキーワード領域が設けられている。このキーワード領域には、所定キーワードが設定されている。そして判定手段が、上記キーワード領域の設定内容が上記所定キーワードと同一か否かを監視している。同一でない場合は判定手段はスタック異常と判定する。

【0010】 このことにより、万一、何等かの原因でスタックがオーバーフロー等により、上記キーワード領域をスタックポインタが指した場合、当然そのアドレスに何等かのデータが書き込まれることになる。この後、例えばスタックポインタが正常なスタック領域を指す状態になったとしても、キーワード領域が書き換えられているという状態は残っている。したがって、後で、そのキーワード領域にアクセスして内容を調べることで、スタックポインタの値の異常が有ったことが判明する。即ちスタックポインタが変化するその瞬間、あるいはその直

(3)

後に調査しなくても、オーバーフロー等の異常の履歴が残存しているので、CPUに過大な負担をかけない範囲でスタックポインタの過去の異常のチェックが可能となる。このように、常時スタックポインタを監視しているのと同等の効果を、CPUに過大な負担をかけることなく、実施することができる。

【0011】上記所定キーワードは、スタックポインタの異常時に書き換えられるコードと偶然一致する可能性はかなり低い。このため、通常用いられるコード以外で有れば、即ち、例えばプログラムに用いられていない値（コード）、正常時にスタック領域に設定される可能性が低い値（コード）あるいは正常時にスタック領域に設定されることが無い値（コード）で有れば、いかなるコードを所定キーワードとして設定しても本発明の目的を達成できる。特に、予め、プログラムのデータも含めた全コードをチェックしておき、そのコード以外のコードを所定キーワードとして設定してもよい。また、プログラム自体を、プログラムの起動時にプログラム内のデータも含めた全コードをチェックして、存在しないコードを検出する処理と、この処理により得られた存在しないコードをキーワード領域に設定する処理とを設けたプログラムとして、プログラム自身で、使用されるコード以外のコードを所定キーワードとして、自動設定するようにしてもよい。

【0012】キーワード領域が設定されるアドレスは、上記スタック領域に隣接するあるいは近傍のアドレスであればよい。例えば、スタック領域の最高点であるアドレスを越えたアドレスでもよい。この場合、スタック領域の最高点であるアドレスの次のアドレスでもよく、それよりも更に先のアドレスでもよい。スタックは、アドレスの高い方から低い方に積まれて行くので、上記スタック領域の最高点を越えたアドレスとは最高点より低いアドレスである。「スタック領域の最高点であるアドレスの次のアドレス」とは、最高点より一つ低いアドレスであり、「それよりも更に先のアドレス」とは、もっと低いアドレスを指している。

【0013】また逆にスタックの起点となるスタートアドレスよりも高いアドレスにキーワード領域を設けてもよい。例えば、スタートアドレス+1のアドレスをキーワード領域としてもよく、それよりも更に高いアドレスをキーワード領域としてもよい。これは逆にアドレスが戻りすぎる異常の検出に有効である。

【0014】また、このキーワード領域も、一つのアドレス領域のみでなく、複数のアドレスにわたってもよく、また連続して存在せずに不連続にキーワード領域を設定してもよい。このようにすることにより、スタックポインタの各種の異常の検出に有効であり、また書き換えられている領域の広さからその異常の程度も判定できる。

【0015】

【実施例】以下、この発明をエンジン制御装置に具体化した実施例について図面に基づいて説明する。

【実施例1】図1は、実施例のエンジン制御装置を示すブロック図である。同図に示すように、制御回路1内に設けられたマイクロコンピュータ3は、CPU（中央処理装置）5、リードオンリメモリ（ROM）7、ランダムアクセスメモリ（RAM）9、タイマー11および入出力ポート15を備えている。また、これらの構成要素はバス13を介して接続されており、このバス13を介して入出力データの転送が行われるようになっている。ROM7内には、エンジン制御用プログラム、その他の処理プログラムおよびそれらの演算処理に必要な種々のデータが予め格納されている。タイマー11はフリーランニングカウンタ、アウトプットコンペアレジスタ等を有する。なお、図1には示していないが、マイクロコンピュータ3は、周知の入出力制御回路、メモリ制御回路等を備えている。

【0016】また、制御回路1において、エアフローメータ、負圧センサ、水温センサ等のセンサ群17からの電圧信号は、アナログマルチプレクサを含むA/D変換器19に送り込まれる。そして、その後前記電圧信号がA/D変換器19にて所定の変換周期で順次2進の信号に変換され、入出力ポート15に送り込まれる。また、スタータスイッチ、スロットルセンサ等SW群21からの信号は入力バッファ23に入力された後、入出力ポート15に送り込まれる。基準位置センサ、クランク角センサ等タイミングセンサ群25からの角度位置信号は、波形整形回路27で矩形形状の基準位置信号及びクランク角信号に波形整形された後、入出力ポート15に送り込まれる。

【0017】このようにして取り込まれたデータに基づいて、プログラムによりCPU5にて、所定の演算処理が実行される。この演算結果に基づいて生成された出力データが、入出力ポート15、出力回路31を介してアクチュエータ群29に送り込まれる。このことにより所望のエンジン制御が実現されている。

【0018】CPU5は図2に示すごとく、スタックポインタ33を含むレジスタ群35を有する。スタックポインタ33は図3に示すごとく、RAM9内に割り当てられたスタック領域37内のいずれかのアドレスを指定する。CPU5は必要に応じてスタックポインタ33で示されている位置のスタック領域37からデータを出し入れする。

【0019】制御回路1は上述した構成以外に、図4に示すCPU5の動作状態をモニタする監視部39を有する。この監視部39はCPU5が何らかの理由で動作停止する、もしくは無限ループに入るとWDCパルスを出力できなくなるので、それを検出した監視部39がCPU5を強制リセットする。このような監視部39はウォッチドッグタイマとして一般的に知られている。

(4)

【0020】次に、CPU5により実行される制御を、図5のフローチャートに基づいて説明する。イグニッションスイッチ（IGP）オンによるリセット後に開始されるメインループのイニシャル処理内で、スタックポインタ33初期化により、スタックポインタ33にはスタートアドレス（図3のAs0の位置）が設定される（ステップ101）。その後、図3に示すごとく、本プログラム内でスタックポインタ33が進む最高点（スタック領域で最低アドレス）Asnの次のアドレスA0、すなわちスタックポインタ33が到達し得ないアドレスにキーワードKを書き込む（ステップ103）。

【0021】ここでキーワードKは、プログラム内容を調べてスタック領域37に書かれない値を設定しておく。具体的には割込処理、サブルーチン・コール時の戻り先アドレス、及び回避データ、またプログラム実行時にワークRAMとしてスタック領域に格納したデータが取り得ない値である。この様な値が存在しない場合は、出現頻度が最低のデータ値を設定する。このようにプログラム内容を調べる処理は、予めプログラム設計時に完了してプログラム自体にそのデータを組み込んでおいてもよい。またプログラムの内容を調べて、スタック領域に書かれることがない値を設定する、あるいは頻度が最低の値を設定する処理をプログラム自体に組み込んでおき、キーワードKの書き込み処理（ステップ103）の前に実行して、適切なキーワードKを決定できるようにしてもよい。

【0022】上述したイニシャル処理（ステップ101、103等）の終了後、メイン処理105のループに入る。メイン処理105は複数のサブルーチン（Sb1、Sb2、…）を含んで構成される。また、メイン・ループ処理中にタイマー11等が割り込み処理（Int1、…）を起動する。またサブルーチン処理や割り込み処理中には図5に示すごとくネスティングが存在する。

【0023】図7を参照して、スタックポインタ33の動作例を説明する。イニシャル終了後、またはメイン処理が次の周回に入った時、正常で有れば、当然、スタックポインタ33はスタック領域のスタート・アドレスAs0を指す。そしてメイン処理105の中でサブルーチンSb1がコールされた時、コール元アドレスをスタート・アドレスAs0に格納してスタックポインタ33は次のアドレスAs1を指す（ステップ201）。尚、本実施例では16ビットCPUの処理なのでスタックポインタはアドレスが2つつ変化される。

【0024】その後、割り込みが発生して割り込み処理Int1により、割り込まれた元アドレスをアドレスAs1に格納してスタックポインタ33は次のアドレスAs2を指す（ステップ301）。更に、割り込み処理Int1内で各種レジスタの内容をPUSH命令（退避させるレジスタ等の内容をスタックに積む命令）を実行する（ステップ303）ため、スタックポインタ33はAs2

からAs3に移動する。その後、割り込み処理Int1内からサブルーチンSb2がコールされて（ステップ305）、スタックポインタ33はAs4を指す（ステップ401）。その後、サブルーチンSb2の終了時にスタックポインタ33はAs4からAs3に戻り（ステップ403）、更に、割り込み処理Int1で、PULL命令（退避されていたレジスタ等の内容を復帰させる命令）が実行されて、スタックポインタ33はAs3からAs2に戻り（ステップ307）、更に割り込み処理Int1終了時にスタックポインタ33はAs2からAs1に戻る（ステップ309）。更に、サブルーチンSb1に戻った後、その終了時にスタックポインタ33はAs1からAs0に戻り（ステップ203）、メイン処理に帰って来る。

【0025】このような処理を繰り返している間に、スタックポインタ33の値がいくつか余分に進んでしまう異常が発生することがある。例えば、プログラム動作中にノイズ等何らかの理由によりサブルーチン、割り込み処理がコールされる前の状態に復帰しない様な誤動作が発生したり、上述したPUSH命令後PULL命令が実施されなかった様な場合である。このような事態が生じると、その時点あるいはそれ以後のスタックポインタ33の変更処理でスタック領域37の最高点であるアドレスAsnを越える場合がある。このような場合には、アドレスAsnの一つ下のキーワード領域A0をスタックポインタ33が指すことになり、この状態で割込処理、サブルーチンコールあるいはデータの退避処理が実行されると、キーワード領域A0に帰る先のアドレスや回避データが書き込まれてしまい、キーワードKは破壊されてしまう。

【0026】この状態は図6に示すスタック異常検出処理にて、破壊されたか否かが判定されて、破壊されている場合は必要な処理が実行される。スタック異常検出処理はメイン処理の1つであり8ms毎にコールされるサブルーチンとし、イニシャル時にステップ103の処理で書き込まれたキーワードKと現在のキーワード領域A0の内容と異なるかどうかを判別する（ステップ501）。異なる場合はこのまま元の処理に復帰する。異なる場合はWDC信号のLオセットを無限に繰り返す処理（ステップ503）に移る。即ち、図4に示したWDC信号が低レベルに固定されてしまうことになる。このようにCPU5は正常にWDCパルスを出力できないため、監視部39がこの状態を検出してCPU5を強制リセットする。従って、CPU5は再度立ち上げ直されて正常な状態で処理を開始することになる。

【0027】このように図6の処理により、キーワード領域A0より低いアドレスにスタックポインタ33が進んで、サブルーチンがコール元に戻らずに暴走したり、RAM領域が破壊されることを防止できる。しかもハードウェア手段を必要とせず瞬間的なスタック異常が生じていても、キーワード領域A0にキーワードKの破壊と

して残存しているのでその検出が可能となった。

【他の実施例】実施例2として、キーワード領域設定の他の例を示す。図3にAxで示すように、キーワード領域のアドレス（16ビット分）も一つでなく、2つでもそれ以上でも良い。このようにキーワード領域を複数のアドレスに設定し、その全領域をステップ501でチェックして、そのアドレスの一つでも破壊されているキーワードが発見されれば、ステップ503を実行すれば良い。こうすれば、万一、領域A0を飛び越すようなスタックポインタ33の異常が生じてても発見することができ、異常発見の確率が増大する。例えば、いかなる値をもスタック領域の値が取り得る場合等に発見精度を上げることができる。

【0028】実施例3として、スタックポインタ33が進む最高点（スタック領域中の最低アドレス）の次のアドレスにキーワードKを置いているが、図3にAyで示すように、スタックポインタ33のスタートアドレス+1のアドレスにもキーワードKを置いても良い。更にこのようにスタック領域37よりも高い方のアドレスにキーワード領域を配置する場合も、複数アドレスにわたってキーワード領域を配置しても良く、それらをステップ501でチェックして、そのアドレスの一つでも破壊されているキーワードKが発見されれば、ステップ503を実行すれば良い。こうすれば、逆にアドレスが戻りすぎる異常の検出に有効である。

【0029】実施例4として、実施例1、2、3を組み合わせたキーワード領域の配置としても良い。このようにすれば、より一層異常発見の制度が向上する。尚、各実施例において、図6のチェックプログラムを8ms毎の割込みとしているがこれに限定されない。チェックプログラムはメインルーチンに直接配置してもよく、更に、CPU5の負担が過大にならない限りサブルーチンや割込ルーチンに配置しても良い。また、CPU5が割り込んでエンジン制御に影響のないタイミングを選択し、割り込みにて図6の処理を実行してもよい。

【0030】また、キーワードKとしては、異常時にキーワードKと同じ値（コード）がキーワード領域に書き込まれるのを防止するため、正常時にスタック領域37に設定される可能性が低い値あるいは正常時にスタック*

*領域37に設定されることが無い値が設定される。このようなキーワードKを決定する場合、16ビット（2バイト）のキーワードKそのものの設定が行われない可能性をチェックしても良いが、8ビット（1バイト）としてスタック領域37に設定される可能性の低いあるいは無いコードを決定し、それを2つ並べて、16ビットのキーワードKとしても良い。また、その8ビットを反転して、反転前の8ビットと反転後の8ビットとを並べて、16ビットのキーワードKとしても良い。また、8ビット自体がスタック領域37に設定される可能性が低いまたは無いならば、これに他のいかなる8ビットを組み合わせて構成した16ビットでもよい。またその8ビットは、スタック領域37に設定される可能性が低いまたは無いのであるから、その8ビットに対して、その8ビットに所定の演算をして求めた8ビット、あるいは予め決定しておいた他の適当な8ビットコードを組み合わせ16ビットのキーワードKとしても良い。

【0031】上記各実施例において、ステップ501の処理が判定手段としての処理に該当する。CPU5として、16ビットタイプを用いたが、8ビットタイプ、32ビットタイプ等のいずれを用いてもよい。

【図面の簡単な説明】

【図1】 実施例のエンジン制御装置を示すブロック図である。

【図2】 レジスタ群の構成説明図である。

【図3】 RAM内のスタック領域周辺の説明図である。

【図4】 CPUとCPUの動作状態をモニタする監視部との関係を示すブロック図である。

【図5】 CPUが実施する処理の一例を示すフローチャートである。

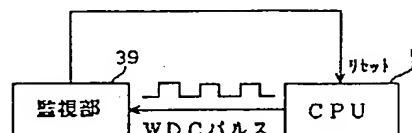
【図6】 スタック異常検出処理のフローチャートである。

【図7】 スタックポインタの動作例の説明図である。

【符号の説明】

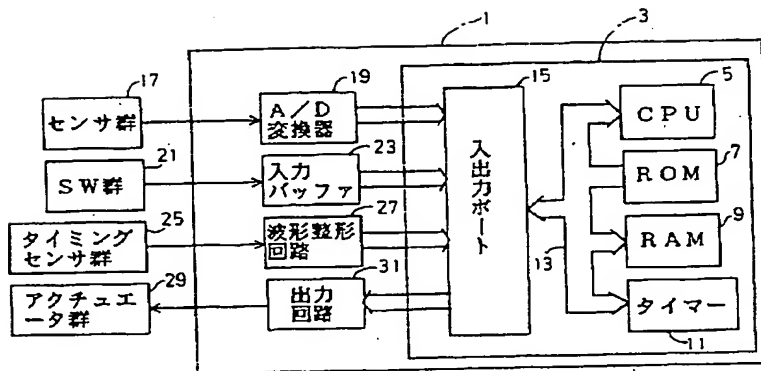
A0…キーワード領域	1…制御回路
3…マイクロコンピュータ	5…CPU
33…スタックポインタ	35…レジスタ群
37…スタック領域	39…監視部

【図4】

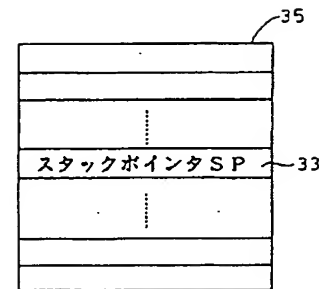


(6)

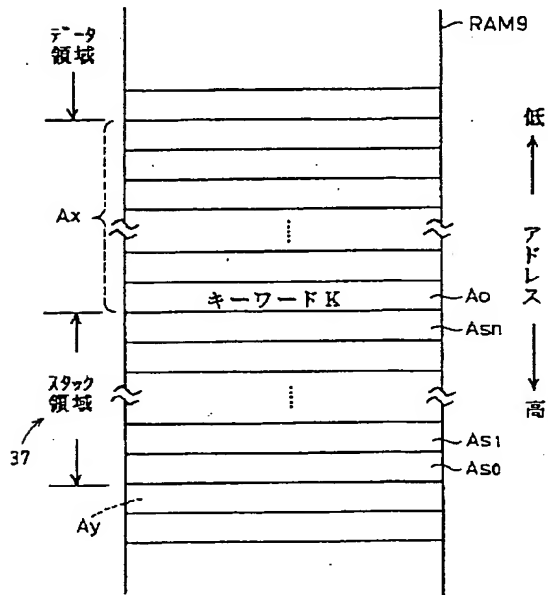
【図1】



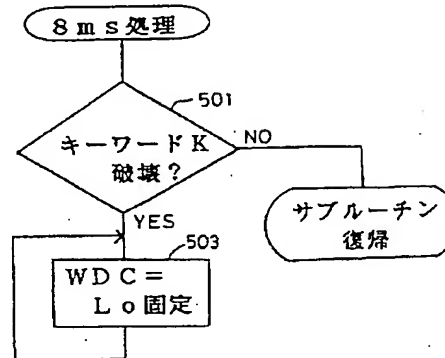
【図2】



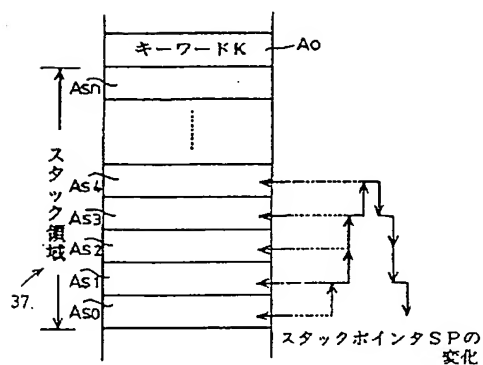
【図3】



【図6】

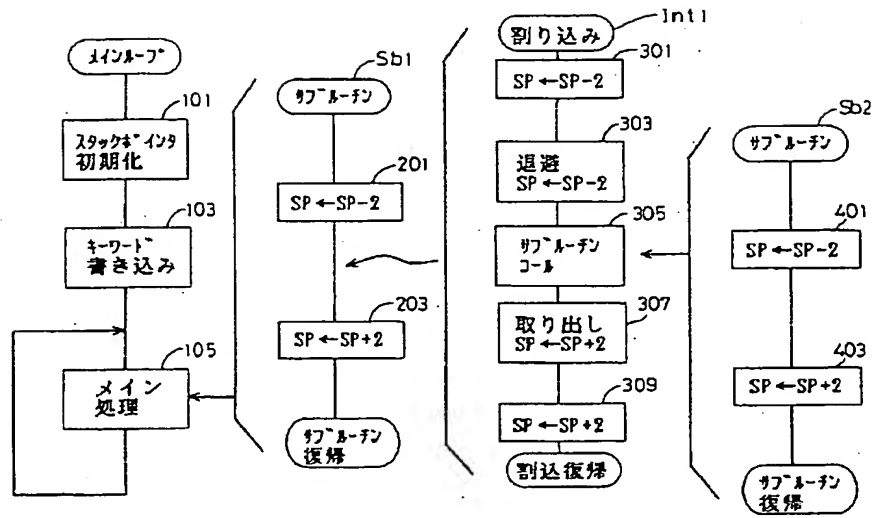


【図7】



(7)

【図5】



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☒ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☒ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.